Here I'll be putting down random ideas, many of which haven't been thought through.

Idea for improving backtracking: maintain a search stack for each connected component of the problem. The order of decisions between stacks will probably need to be maintained, because once two connected components get merged, you might need to backjump arbitrarily far into either stack. I'm not sure if this will present any sort of actual benefit. Should probably look at how connected components evolve over the course of our problems.

Distance heuristic: I'll have to go back and re-read the Dave and Sarah's paper, but I think they only apply the heuristic in the forward direction. I wonder if it's possible to apply this heuristic to non-guard parameter decisions. I think it would be quite difficult to do. Guard bindings and open conditions/threats have fairly predictable, easily observable effects on the plan, but non-guard parameter bindings mostly affect future merge decisions.

You can tell which parameters belong to which token by their entity keys. Currently:

```
token = x
state = x + 1
object = x + 2
duration = x + 3
start = x + 5
end = x + 6
parameters start at x + 7
```

Note that x + 4 is the ObjectTokenRelation constraint.